

	N. Aillery
	C. Marais
	Orange
	August 30, 2016

OpenID Connect User Questioning API 1.0

draft-user-questioning-api-01

Abstract

This specification defines a specific endpoint used by a Client (i.e. Service Provider) in order to question a End-User and get his Statement (i.e. his answer).

Table of Contents

1. Introduction
 - 1.1. Requirements Notation and Conventions
 - 1.2. Terminology
 - 1.3. Overview
 - 1.4. Example of use cases involving the User Questioning API
2. Question Object
 - 2.1. error_info SubObject
 - 2.2. error_code
 - 2.3. status
 - 2.4. user_id_type
3. User Questioning flows
 - 3.1. Pulled-By-Client Flow
 - 3.1.1. Pulled-By-Client Flow steps
 - 3.1.2. User Questioning Endpoint
 - 3.1.2.1. (1a) User Questioning Request
 - 3.1.2.1.1. Example with an access_token tied with a specific End-User
 - 3.1.2.1.2. Example with an access_token NOT tied with a specific End-User
 - 3.1.2.2. (1b) Question Created
 - 3.1.2.2.1. Example with an access_token tied with a specific End-User
 - 3.1.2.2.2. Example with an access_token not tied with a specific End-User
 - 3.1.2.3. (2a) User interactions to get the Questioned User's Statement
 - 3.1.2.4. (3a) Get User Questioning Response
 - 3.1.2.4.1. Example not using If-None-Match header
 - 3.1.2.4.2. Example using If-None-Match header
 - 3.1.2.5. (3b) User Questioning Response
 - 3.1.2.5.1. Example with PENDING status and no If-None-Match header in the User Questioning Request
 - 3.1.2.5.2. Example with PENDING status and a If-None-Match header in the User Questioning Request
 - 3.1.2.5.3. Example with ACCEPTED status
 - 3.1.2.5.4. Example with DENIED status
 - 3.2. Pushed-To-Client Flow
 - 3.2.1. Pushed-To-Client Flow Steps
 - 3.2.2. User Questioning Endpoint
 - 3.2.2.1. (1a) User Questioning Request
 - 3.2.2.1.1. Example with an access_token tied with a specific End-User
 - 3.2.2.1.2. Example with an access_token NOT tied with a specific End-User
 - 3.2.2.2. (1b) Question Created
 - 3.2.2.2.1. Example with an access_token tied with a specific End-User
 - 3.2.2.2.2. Example with an access_token not tied with a specific End-User
 - 3.2.2.3. (2a) User interactions to get the Questioned User's Statement
 - 3.2.3. Client Notification Endpoint
 - 3.2.3.1. (3a) User Questioning Response
 - 3.2.3.1.1. Example with ACCEPTED status
 - 3.2.3.1.2. Example with DENIED status
 - 3.2.3.2. (3b) Acknowledgement
 - 3.2.3.2.1. Example of the simplest acknowledgement
 - 3.3. Terminated-By-Client Flow
 - 3.3.1. Terminated-By-Client Flow steps
 - 3.3.2. User Questioning Endpoint
 - 3.3.2.1. (1a) User Questioning Request
 - 3.3.2.1.1. Example with an access_token tied with a specific End-User
 - 3.3.2.1.2. Example with an access_token NOT tied with a specific End-User

3.3.2.2. (1b) Question Created

3.3.2.2.1. Example with an access_token tied with a specific End-User

3.3.2.2.2. Example with an access_token not tied with a specific End-User

3.3.2.3. (2a) User interactions to get the Questioned User's Statement

3.3.2.4. (2b) User Questioning Endpoint Sends Verification_code to Questioned User

3.3.2.5. (2c) Questioned User Sends Verification_code to Client

3.3.2.6. (3a) Client Sends Verification_code to User Questioning Endpoint

3.3.2.6.1. Example

3.3.2.7. (3b) User Questioning Response

3.3.2.7.1. Example of Accepted Statement

3.3.2.7.2. Example of Denied Statement

4. Errors

4.1. Example with ERROR status in a response to Client

4.2. Example with ERROR status in a request to Client

5. Security Considerations

6. Privacy Considerations

7. IANA Considerations

8. Normative References

Appendix A. Acknowledgements

Appendix B. Notices

Appendix C. Document History

Authors' Addresses

1. Introduction

This specification defines a specific endpoint used by a Client (i.e. Service Provider) in order to question a End-User and get his Statement (i.e. his answer).

This endpoint is specified as an OAuth 2.0-protected Resource Server accessible with an Access Token.

The way the Access Token has been obtained by the Client is out of scope of this specification.

Whether the End-User is currently using the Client or not is also out of scope of this specification.

The User Questioning API is an asynchronous API. There are 3 main ways to get the End-User's Statement: the first one requires some polling of the API, the second requires the Client to expose a callback endpoint and the third one requires a user interaction on the Client.

1.1. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value.

1.2. Terminology

This specification uses the terms "Access Token", "Resource Server", and "Client" defined by [OAuth 2.0 \[RFC6749\]](#). This specification also defines the following terms:

Questioned User

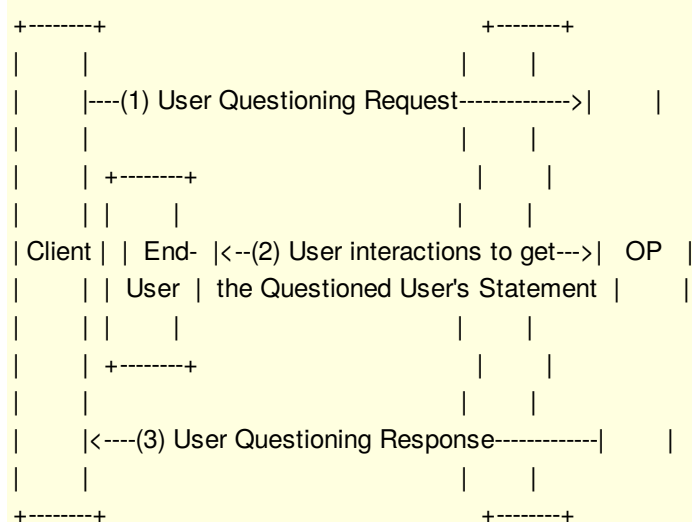
End-User receiving the question and requested to give his Statement.

1.3. Overview

The User Questioning protocol, in abstract, follows the following steps.

1. The Client sends a User Questioning Request to the OpenID Provider (OP).
2. The OP interacts with the Questioned User and obtains his Statement.
3. The OP responds to the Client with a User Questioning Response.

These steps are illustrated in the following diagram:



1.4. Example of use cases involving the User Questioning API

The following use cases are non-normative examples to illustrate the usage of the User Questioning API by a Client.

1. The Client can be a bank and the User Questioning API is used to challenge the End-User when he wants to pay on Internet in order to secure the transaction.. This is similar to 3D-Secure. The question could be: "Do you allow a payment of XXX euros to XXX?".
2. The Client can be a bank and the User Questioning API is used to challenge the End-User when he wants to add a new beneficiary for a bank transfer. The question could be: "Do you allow XXX to be added to your beneficiaries?".
3. The Client can be a drive-in food market and the User Questioning API is used to ask the End-User if he accepts the exchange of one missing product by another. The question could be: "Do you agree to get product XXX instead of product XXX?".
4. The Client can be a ticketing platform and the User Questioning API is used to prevent transactions by bots. The question could be: "Do you confirm that you are currently booking a ticket?".
5. The Client can be an airline company and the User Questioning API is used to be sure that the End-User is notified of a delay. The question could be: "Your flight is postponed. Can you confirm that you are aware?".

2. Question Object

Here after is described the Question object :

Member	Type	Description	Example
id	string	Unique identifier of the Question	984dcc7d-3d4d-4b0f-9f80-22e344f9a956
status	string	Status code of the Question (note that Error codes are handled by the HTTP protocol) Possible values are described in Section 2.3 .	PENDING
creation_date	timestamp	Date indicating when the User Questioning Request has been received by the OP. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.	1311281970
last_modification_date	timestamp	Date indicating the last change of the status. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.	1311281970
statement_date	timestamp	Date indicating when the End-User gave his Statement on the Question. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.	1311282970
user_id	string	Unique identifier allowing to identify the Questioned User (e.g. Mobile phone , PCR). Ignored if the Access Token is tied with a specific End-User, mandatory otherwise.	3444975f-1137-47dc-908f-942ac85ab98f
user_id_type	string	Indicate the type of the End-User's identifier used for User Questioning. Ignored if the Access Token is tied with a specific End-User, mandatory otherwise. Possible values are described in Section 2.4 .	MSISDN
question_to_display	string	Question to be displayed to the Questioned User.	An example message to display
wished_qcr	string	Questioning context class reference : Level of Assurance wished by the Client (can be 2 3 4)	3
used_qcr	string	Questioning context class reference : Level of Assurance used by the OP (can be 2 3 4)	2
wished_qmr	string	Questioning method wished by the Client	SMS_OTP
used_qmr	string	Questioning method used by the OP for the User Questioning	SMS_OTP

Member	Type	Description	Example
client_notification_endpoint	uri	URL exposed by the Client's backend to receive a notification from OP when a User Questioning is finished (accepted, denied, error...). Needed if the Client wants to use Pushed-To-Client Flow (See Section 3.2).	https://client.example.com/questions
verification_code	string	Unique code collected by the Client and verified by the OP to allow implementing a One Time Password as a User Questioning method (Optional: only needed if OTP is used as a validator. So only if the reponse to POST of an User Questioning Request was responded by a Question Object with a status=VERIFICATION_CODE_NEEDED.)	12345
error_info	JSON Object	(Optional). Only present if status is ERROR. See Section 2.1 .	{"error_code":"unknown_user"}

Question Object Members

2.1. error_info SubObject

error_info is an optional member of the Question Object which contains the following properties :

Value	Description	Example
error_code	REQUIRED. Code representing the error. See section Section 2.2 for possible values.	unknown_user
error_description	OPTIONAL. Human-readable ASCII encoded text description of the error.	The user is unknown
error_uri	OPTIONAL. URI of a web page that includes additional information about the error.	https://server.example.com/errors/unknown_user

error_info properties

2.2. error_code

error_code is a property of the error_info Object. It can takes the following values :

Value	Description
unknown_user	The Questioned User is unknown.
timeout	The User Questioning has expired (timeout - no action from Questioned User)
verification_code_failed	The verification_code provided was not correct.
verification_code_too_many_tries	The verification_code has already been check without success several times (the number of time is up to the OP).

error_code possible values

2.3. status

status member can take the following values :

Value	Description
ACCEPTED	This status name is returned when the Questioned User's Statement is an acceptance.
DENIED	This status name is returned when the Questioned User's Statement is a deny.
ERROR	This status name is returned when an error is detected by the OP. (Error number is provided by the HTTP return code.). The error_info member will give more details concerning the error See Section 2.1 .
PENDING	This status name is returned when the User Questioning is ongoing.
VERIFICATION_CODE_NEEDED	This status name is returned when a verification_code is required to be requested by the Client to the Questioned User.

status possible values

2.4. user_id_type

user_id_type member can take the following values :

Value	Description	Example
MSISDN	Represents the Mobile Phone number corresponding to the Questioned User. The way this MSISDN has been captured by the Client is out of scope of this specification.	33612345678
PCR	Represents the PCR corresponding to the Questioned User. The way this PCR has been captured by the Client is out of scope of this specification.	8d858e0a-c91b-426a-92e8-462d3876df7d

user_id_type possible values

3. User Questioning flows

This document specifies three User Questioning flows:

Pulled-By-Client flow:

In this flow, after the User Questioning Request, the Client must call the OP in order to get the User Questioning Response. Refer to [Section 3.1](#) for more details.

Pushed-To-Client flow:

In this flow, after the User Questioning Request, the OP calls the Client to deliver the User Questioning Response. Refer to [Section 3.2](#) for more details.

Terminated-By-Client flow:

In this flow, after the User Questioning Request, the Client must call the OP with an additional verification_code in order to get the User Questioning Response. Refer to [Section 3.3](#) for more details.

The flow to use is decided by the OP and depends on the User Questioning Request and the available User Questioning mechanisms:

Pulled-By-Client flow:

If the Client does not include a client_notification_endpoint in the User Questioning Request and if the User Questioning mechanism selected by the OP does not require additional information from the Client (e.g. verification_code). Refer to [Section 3.1](#) for more details.

Pushed-To-Client flow:

If the Client includes a client_notification_endpoint in the User Questioning Request and if the User Questioning mechanism selected by the OP does not require additional information from the Client (e.g. verification_code). Refer to [Section 3.2](#) for more details.

Terminated-By-Client flow:

If the User Questioning mechanism selected by the OP requires additional information from the Client

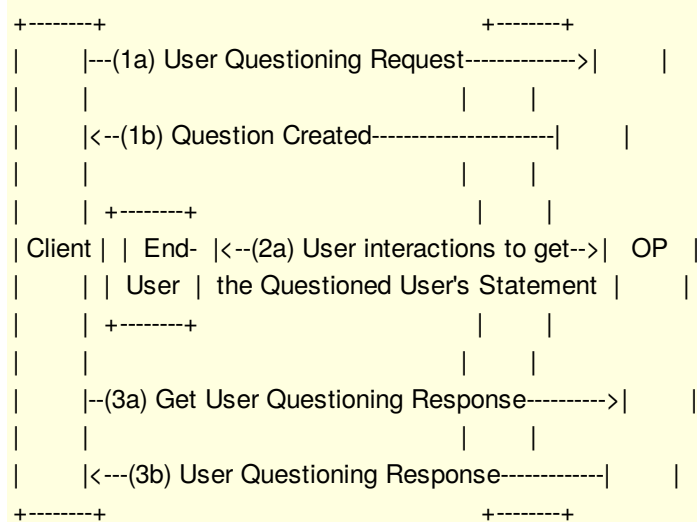
(e.g. verification_code). Refer to [Section 3.3](#) for more details.

3.1. Pulled-By-Client Flow

In this flow, the Client MUST NOT include a client_notification_endpoint in the User Questioning Request. The Client will poll the OP until the User Questioning is finished (accepted, denied, error...).

If a verification_code is needed for the mechanism chosen by the OP, the flow will be the Terminated-By-Client Flow (cf. [Section 3.3](#)).

3.1.1. Pulled-By-Client Flow steps



3.1.2. User Questioning Endpoint

3.1.2.1. (1a) User Questioning Request

The Client sends the User Questioning Request using HTTP POST.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
user_id	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
client_notification_endpoint	FORBIDDEN
...any other...	IGNORED

3.1.2.1.1. Example with an access_token tied with a specific End-User

This example describes a context where the access_token is tied with a specific End-User. The Client MAY add user_id and user_id_type in the request, but they will be ignored.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SIAV32hkKG

{
  "question_to_display": "An example message to display",
  "wished_qcr": "3"
}
```

3.1.2.1.2. Example with an access_token NOT tied with a specific End-User

This example describes a context where the access_token is not tied with a specific End-User. The Client MUST add user_id and user_id_type in the request.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SIAV32hkKG

{
  "user_id": "33612345678",
  "user_id_type": "MSISDN",
  "question_to_display": "An example message to display",
  "wished_qcr": "3"
}
```

3.1.2.2. (1b) Question Created

The Client receives the created Question object in a HTTP 201 Created response.

The Question object is contained in the response body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
...any other...	FORBIDDEN

In this flow, there are three possible status (cf. [Section 2.3](#)) in the Question object received:

```
{"status":"PENDING"}
```

This is the temporary status when the User Questioning Response is not ready.

```
{"status":"VERIFICATION_CODE_NEEDED"}
```

This is a temporary status when an additional verification_code is required.

If the status is VERIFICATION_CODE_NEEDED, then the current flow is the Terminated-By-Client Flow ([Section 3.3](#)).

```
{"status":"ERROR"}
```

This is the final status when there is an error. Refer to [Section 4](#) for more details.

3.1.2.2.1. Example with an access_token tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"PENDING",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "question_to_display":"An example message to display",
  "wished_qcr":"3"
}
```

3.1.2.2.2. Example with an access_token not tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"PENDING",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "user_id":"33612345678",
  "user_id_type":"MSISDN",
  "question_to_display":"An example message to display",
  "wished_qcr":"3"
}
```

3.1.2.3. (2a) User interactions to get the Questioned User's Statement

The way the User Questioning Endpoint obtains the Questioned User's Statement for the Question is out of the scope of this specification.

3.1.2.4. (3a) Get User Questioning Response

The Client get the User Questioning Response using HTTP GET.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The Question object is associated with an Etag. When the object is modified, the associated Etag MUST be modified too. In order to detect a status change, the Client CAN either request the Question object or, by using the If-None-Match, request the modified Question object.

The If-None-Match header contains the Etag. If the Question object has not been modified, the OP responds with HTTP 304 code.

3.1.2.4.1. Example not using If-None-Match header

If the User Questioning Response is not finished, not using the If-None-Match header enables to receive a HTTP 200 code with the current Question object including a PENDING status.

The following is a non-normative example.

```
GET /questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c HTTP/1.1
Host: server.example.com
Accept: application/json
Authorization: Bearer SIAV32hkKG
```

3.1.2.4.2. Example using If-None-Match header

If the User Questioning Response is not finished, the If-None-Match header enables to receive a HTTP 304 code, instead of the current Question object including a PENDING status.

The following is a non-normative example.

```
GET /questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c HTTP/1.1
Host: server.example.com
Accept: application/json
Authorization: Bearer SIAV32hkKG
If-None-Match: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"
```

3.1.2.5. (3b) User Questioning Response

The Client receives the created User Questioning Response in a HTTP 200 OK response.

The User Questioning Response MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
statement_date	FORDIDDEN if status is PENDING, MANDATORY if the status is ACCEPTED or DENIED.
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User

Attribute name	Presence
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
used_qcr	FORDIDDEN if status is PENDING or ERROR, MANDATORY if the status is ACCEPTED or DENIED.
used_qmr	FORDIDDEN if status is PENDING or ERROR, MANDATORY if the status is ACCEPTED or DENIED.
...any other...	FORBIDDEN

In this flow, there are four possible status (cf. [Section 2.3](#)) in the Question object received:

```
{"status":"PENDING"}
```

This is the temporary status when the User Questioning Response is not ready.

```
{"status":"ACCEPTED"}
```

This is the final status when the End-User accepted the User Questioning Request.

```
{"status":"DENIED"}
```

This is the final status when the End-User denied the User Questioning Request.

```
{"status":"ERROR"}
```

This is the final status when there is an error. Refer to [Section 4](#) for more details.

If the If-None-Match header was used in the User Questioning Request, instead of a HTTP 200 code with the current Question object including a PENDING status, the Client will get a HTTP 304 code and no body.

3.1.2.5.1. Example with PENDING status and no If-None-Match header in the User Questioning Request

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"PENDING",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "question_to_display":"An example message to display",
  "wished_qcr":"3"
}
```

3.1.2.5.2. Example with PENDING status and a If-None-Match header in the User Questioning Request

The following is a non-normative example.

3.1.2.5.3. Example with ACCEPTED status

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "5a530eac19fc-9deb-a1ae-49ba-442c8ae3"

{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "ACCEPTED",
  "creation_date": "1311281970",
  "last_modification_date": "1311282970",
  "statement_date": "1311282970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "used_qcr": "2",
  "used_qmr": "SMS_OTP"
}
```

3.1.2.5.4. Example with DENIED status

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "5a530eac19fc-9deb-a1ae-49ba-442c8ae3"

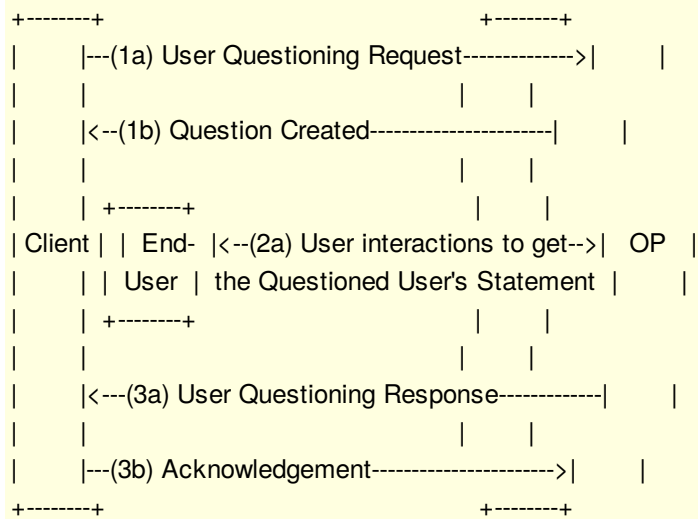
{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "DENIED",
  "creation_date": "1311281970",
  "last_modification_date": "1311282970",
  "statement_date": "1311282970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "used_qcr": "2",
  "used_qmr": "SMS_OTP"
}
```

3.2. Pushed-To-Client Flow

In this flow, the Client **MUST** include a `client_notification_endpoint` in the User Questioning Request. The Client will be informed at this endpoint by the OP when the User Questioning is finished (accepted, denied, error...).

If a `verification_code` is needed for the mechanism chosen by the OP, the flow will be the Terminated-By-Client Flow (cf. [Section 3.3](#)), despite the `client_notification_endpoint` in the request.

3.2.1. Pushed-To-Client Flow Steps



3.2.2. User Questioning Endpoint

3.2.2.1. (1a) User Questioning Request

The Client sends the User Questioning Request using HTTP POST.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
user_id	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
client_notification_endpoint	MANDATORY
...any other...	IGNORED

3.2.2.1.1. Example with an access_token tied with a specific End-User

This example describes a context where the access_token is tied with a specific End-User. The Client MAY add user_id and user_id_type in the request, but they will be ignored.

The following is a non-normative example.

```

POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SIAV32hkKG

{
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "client_notification_endpoint": "https://server.client.com/questions"
}
  
```

```
}
```

3.2.2.1.2. Example with an access_token NOT tied with a specific End-User

This example describes a context where the access_token is not tied with a specific End-User. The Client MUST add user_id and user_id_type in the request.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SIAV32hkKG

{
  "user_id": "33612345678",
  "user_id_type": "MSISDN",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "client_notification_endpoint": "https://server.client.com/questions"
}
```

3.2.2.2. (1b) Question Created

The Client receives the created Question object in a HTTP 201 Created response.

The Question object is contained in the response body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
client_notification_endpoint	MANDATORY
...any other...	FORBIDDEN

In this flow, there are three possible status (cf. [Section 2.3](#)) in the Question object received:

```
{"status": "PENDING"}
```

This is the temporary status when the User Questioning Response is not ready.

```
{"status": "VERIFICATION_CODE_NEEDED"}
```

This is a temporary status when an additional verification_code is required.

If the status is VERIFICATION_CODE_NEEDED, then the current flow is the Terminated-By-Client Flow

([Section 3.3](#)).

```
{"status":"ERROR"}
```

This is the final status when there is an error. Refer to [Section 4](#) for more details.

3.2.2.2.1. Example with an access_token tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"PENDING",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "question_to_display":"An example message to display",
  "wished_qcr":"3",
  "client_notification_endpoint":"https://server.client.com/questions"
}
```

3.2.2.2.2. Example with an access_token not tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"PENDING",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "user_id":"33612345678",
  "user_id_type":"MSISDN",
  "question_to_display":"An example message to display",
  "wished_qcr":"3",
  "client_notification_endpoint":"https://server.client.com/questions"
}
```

3.2.2.3. (2a) User interactions to get the Questioned User's Statement

The way the User Questioning Endpoint obtains the Questioned User's Statement for the Question is out of the scope of this specification.

3.2.3. Client Notification Endpoint

3.2.3.1. (3a) User Questioning Response

The OP sends the User Questioning Response to the Client using HTTP POST.

The User Questioning Response MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
statement_date	MANDATORY
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
used_qcr	MANDATORY
used_qmr	OPTIONAL
client_notification_endpoint	MANDATORY
...any other...	IGNORED

In this flow, there are three possible status (cf. [Section 2.3](#)) in the User Questioning Response:

```
{"status": "ACCEPTED"}
```

This is the final status when the End-User accepted the User Questioning Request.

```
{"status": "DENIED"}
```

This is the final status when the End-User denied the User Questioning Request.

```
{"status": "ERROR"}
```

This is the final status when there is an error. Refer to [Section 4](#) for more details.

3.2.3.1.1. Example with ACCEPTED status

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.client.com
Accept: application/json

{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "ACCEPTED",
  "creation_date": "1311281970",
  "last_modification_date": "1311282970",
  "statement_date": "1311282970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "used_qcr": "3",
  "used_qmr": "SIM_APPLET",
  "client_notification_endpoint": "https://server.client.com/questions"
```

```
}
```

3.2.3.1.2. Example with DENIED status

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.client.com
Accept: application/json

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"DENIED",
  "creation_date":"1311281970",
  "last_modification_date":"1311282970",
  "statement_date":"1311282970",
  "question_to_display":"An example message to display",
  "wished_qcr":"3",
  "used_qcr":"3",
  "used_qmr":"SIM_APPLET",
  "client_notification_endpoint":"https://server.client.com/questions"
}
```

3.2.3.2. (3b) Acknowledgement

The acknowledgement MUST be a HTTP 200 OK. The HTTP code is the only parameter to consider. The rest of the HTTP response should be ignored.

3.2.3.2.1. Example of the simplest acknowledgement

The following is a non-normative example.

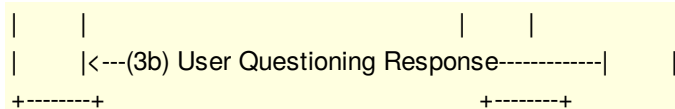
```
HTTP/1.1 200 OK
```

3.3. Terminated-By-Client Flow

In this flow, after the first User Questioning Request, the Client must supply the OP with a `verification_code` in order to finish the User Questioning (accepted, denied, error...).

3.3.1. Terminated-By-Client Flow steps

```
+-----+
| |---(1a) User Questioning Request----->| | | |
| |                                     | |
| |<--(1b) Question Created-----| |
| |                                     | |
| | +-----+                         | |
| | |      |<--(2a) User interactions to get-->| |
| | | End- | the Questioned User's Statement | |
| | | User |                               | |
| Client | | |<----(2b) Verification_code-----| OP |
| | +-----+                         | |
| | |                                     | |
| |<-----+ (2c) Verification_code          | |
| |                                     | |
| |---(3a) Verification_code----->| |
```



3.3.2. User Questioning Endpoint

3.3.2.1. (1a) User Questioning Request

The Client sends the User Questioning Request using HTTP POST.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
user_id	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
client_notification_endpoint	IGNORED
...any other...	IGNORED

3.3.2.1.1. Example with an access_token tied with a specific End-User

This example describes a context where the access_token is tied with a specific End-User. The Client MAY add user_id and user_id_type in the request, but they will be ignored.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SIAV32hkKG

{
  "question_to_display": "An example message to display",
  "wished_qcr": "3"
}
```

3.3.2.1.2. Example with an access_token NOT tied with a specific End-User

This example describes a context where the access_token is not tied with a specific End-User. The Client MUST add user_id and user_id_type in the request.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
```

Authorization: Bearer SIAV32hkKG

```
{
  "user_id": "33612345678",
  "user_id_type": "MSISDN",
  "question_to_display": "An example message to display",
  "wished_qcr": "3"
}
```

3.3.2.2. (1b) Question Created

The Client receives the created Question object in a HTTP 201 Created response.

The Question object is contained in the response body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
client_notification_endpoint	OPTIONAL
...any other...	FORBIDDEN

In this flow, there are three possible status (cf. [Section 2.3](#)) in the Question object received:

```
{"status": "PENDING"}
```

This is the temporary status when the User Questioning Response is not ready.

If the status is PENDING, then the current flow is either the Pulled-By-Client Flow ([Section 3.1](#)) or the Pushed-To-Client Flow ([Section 3.2](#)).

```
{"status": "VERIFICATION_CODE_NEEDED"}
```

This is a temporary status when an additional verification_code is required. This is the normal status for this flow.

```
{"status": "ERROR"}
```

This is the final status when there is an error. Refer to [Section 4](#) for more details.

3.3.2.2.1. Example with an access_token tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: 442c8ae3-a1ae-49ba-9deb-5a530eac19fc
```

```
{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"VERIFICATION_CODE_NEEDED",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "question_to_display":"An example message to display",
  "wished_qcr":"3"
}
```

3.3.2.2.2. Example with an access_token not tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: 442c8ae3-a1ae-49ba-9deb-5a530eac19fc

{
  "id":"84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status":"VERIFICATION_CODE_NEEDED",
  "creation_date":"1311281970",
  "last_modification_date":"1311281970",
  "user_id":"33612345678",
  "user_id_type":"MSISDN",
  "question_to_display":"An example message to display",
  "wished_qcr":"3"
}
```

3.3.2.3. (2a) User interactions to get the Questioned User's Statement

The way the User Questioning Endpoint obtains the Questioned User's Statement for the Question is out of the scope of this specification.

3.3.2.4. (2b) User Questioning Endpoint Sends Verification_code to Questioned User

The way the User Questioning Endpoint sends the verification_code to the Questioned User is out of the scope of this specification.

3.3.2.5. (2c) Questioned User Sends Verification_code to Client

The way the End-User sends the verification_code to the Client is out of the scope of this specification.

3.3.2.6. (3a) Client Sends Verification_code to User Questioning Endpoint

The Client sends the User Questioning Request using HTTP PUT.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
verification_code	MANDATORY
...any other...	IGNORED

3.3.2.6.1. Example

The following is a non-normative example.

```
PUT /questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c HTTP/1.1
Host: server.example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer SIAV32hkKG

{
  "verification_code":"12345"
}
```

3.3.2.7. (3b) User Questioning Response

The Client receives the created User Questioning Response in a HTTP 200 OK response.

The User Questioning Response MUST contain a Question object in the request body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
statement_date	MANDATORY
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_to_display	MANDATORY
wished_qcr	MANDATORY
wished_qmr	OPTIONAL
used_qcr	MANDATORY
used_qmr	OPTIONAL
...any other...	IGNORED

In this flow, there are three possible status (cf. [Section 2.3](#)) in the Question object received:

```
{"status":"ACCEPTED"}
```

This is the final status when the End-User accepted the User Questioning Request.

```
{"status":"DENIED"}
```

This is the final status when the End-User denied the User Questioning Request.

```
{"status":"ERROR"}
```

This is the final status when there is an error. Refer to [Section 4](#) for more details.

3.3.2.7.1. Example of Accepted Statement

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "ACCEPTED",
  "creation_date": "1311281970",
  "last_modification_date": "1311282970",
  "statement_date": "1311282970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "used_qcr": "2",
  "used_qmr": "SMS_OTP"
}
```

3.3.2.7.2. Example of Denied Statement

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "DENIED",
  "creation_date": "1311281970",
  "last_modification_date": "1311282970",
  "statement_date": "1311282970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "used_qcr": "2",
  "used_qmr": "SMS_OTP"
}
```

4. Errors

4.1. Example with ERROR status in a response to Client

This example can occur in the Pulled-By-Client (cf. [Section 3.1](#)), Pushed-To-Client (cf. [Section 3.2](#)) and Terminated-By-Client (cf. [Section 3.3](#)) flows.

The following is a non-normative example.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac0e-36b858ccbd8c
Content-Length: xxx
Etag: "442c8ae3-a1ae-49ba-9deb-5a530eac19fc"

{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "ERROR",
  "creation_date": "1311281970",
  "last_modification_date": "1311281970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "error_info":
  {
    "error_code": "unknown_user",
    "error_description": "The user is unknown",
    "error_uri": "https://server.example.com/errors/unknown_user"
  }
}
```

4.2. Example with ERROR status in a request to Client

This example can occur in the Pushed-To-Client flow (cf. [Section 3.2](#)).

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.client.com
Accept: application/json

{
  "id": "84c1d9d6-62e5-4803-ac0e-36b858ccbd8c",
  "status": "ERROR",
  "creation_date": "1311281970",
  "last_modification_date": "1311282970",
  "statement_date": "1311282970",
  "question_to_display": "An example message to display",
  "wished_qcr": "3",
  "error_info":
  {
    "error_code": "unknown_user",
    "error_description": "The user is unknown",
    "error_uri": "https://server.example.com/errors/unknown_user"
  }
}
```

5. Security Considerations

TBD

6. Privacy Considerations

TBD

7. IANA Considerations

This document makes no requests of IANA.

8. Normative References

[JWT]	Jones, M., Bradley, J. and N. Sakimura, " JSON Web Token (JWT) ", Internet-Draft draft-ietf-oauth-json-web-token, May 2013.
[OpenID.Core]	Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C. and E. Jay, "OpenID Connect Standard 1.0", December 2013.
[OpenID.Discovery]	Sakimura, N., Bradley, J., Jones, M. and E. Jay, "OpenID Connect Discovery 1.0", September 2014.
[OpenID.Registration]	Sakimura, N., Bradley, J. and M. Jones, "OpenID Connect Dynamic Client Registration 1.0", December 2013.
[RFC2119]	Bradner, S., " Key words for use in RFCs to Indicate Requirement Levels ", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.
[RFC2246]	Dierks, T. and C. Allen, " The TLS Protocol Version 1.0 ", RFC 2246, DOI 10.17487/RFC2246, January 1999.
[RFC5246]	Dierks, T. and E. Rescorla, " The Transport Layer Security (TLS) Protocol Version 1.2 ", RFC 5246, DOI 10.17487/RFC5246, August 2008.
[RFC6749]	Hardt, D., " The OAuth 2.0 Authorization Framework ", RFC 6749, DOI 10.17487/RFC6749, October 2012.
[RFC6750]	Jones, M. and D. Hardt, " The OAuth 2.0 Authorization Framework: Bearer Token Usage ", RFC 6750, DOI 10.17487/RFC6750, October 2012.

Appendix A. Acknowledgements

The following have contributed to the development of this specification.

Appendix B. Notices

Copyright (c) 2014 The OpenID Foundation.

The OpenID Foundation (OIDF) grants to any Contributor, developer, implementer, or other interested party a non-exclusive, royalty free, worldwide copyright license to reproduce, prepare derivative works from, distribute, perform and display, this Implementers Draft or Final Specification solely for the purposes of (i) developing specifications, and (ii) implementing Implementers Drafts and Final Specifications based on such documents, provided that attribution be made to the OIDF as the source of the material, but that such attribution does not indicate an endorsement by the OIDF.

The technology described in this specification was made available from contributions from various sources, including members of the OpenID Foundation and others. Although the OpenID Foundation has taken steps to help ensure that the technology is available for distribution, it takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this specification or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any independent effort to identify any such rights. The OpenID Foundation and the contributors to this specification make no (and hereby expressly disclaim any) warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to this specification, and the entire risk as to implementing this specification is assumed by the implementer. The OpenID Intellectual Property Rights policy requires contributors to offer a patent promise not to assert certain patent claims against other contributors and against implementers. The OpenID Foundation invites any interested party to bring to its attention any copyrights, patents, patent applications, or other proprietary rights that may cover technology that may be required to practice this specification.

Appendix C. Document History

[[To be removed from the final specification]]

-01

- Initial draft
- Added ODF Standard Notice

Authors' Addresses

Nicolas Aillery

Orange

E-Mail: nicolas.aillery@orange.com

Charles Marais

Orange

E-Mail: charles.marais@orange.com